

FIG. 1

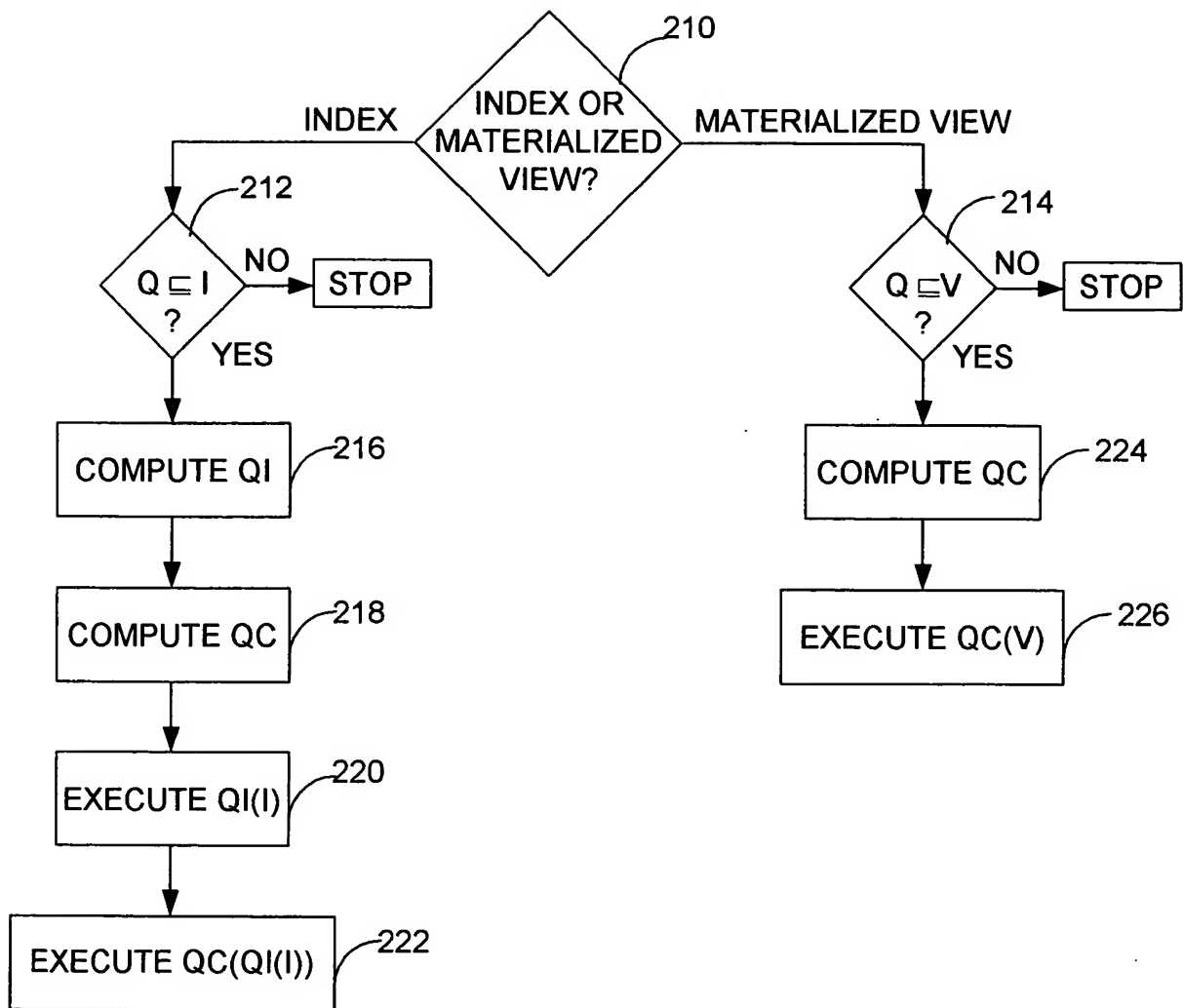


FIG. 2

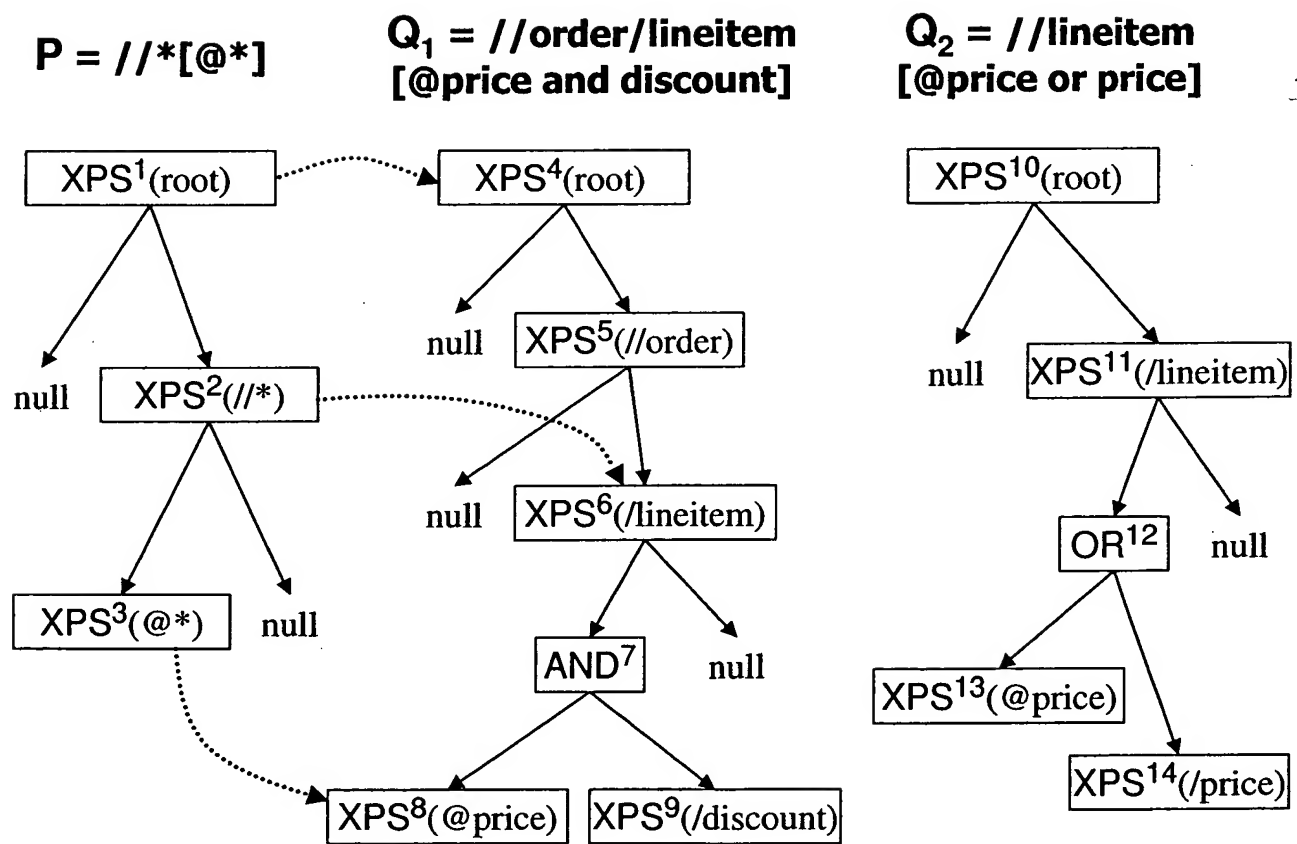


FIG. 3

1	$matchStep(p, q)$	
1.1	if $(q = q_1 \wedge q_2)$	$matchStep(p, q) \rightarrow matchStep(p, q_1) \vee matchStep(p, q_2)$
1.2	else if $(q = q_1 \vee q_2)$	$matchStep(p, q) \rightarrow matchStep(p, q_1) \wedge matchStep(p, q_2)$
1.3	else if $(p_{axis} = \text{"descendant"})$	$matchStep(p, q) \rightarrow \bigvee \{ matchChildren(p, c) \}, \forall c \in \{ \text{preorder traversal of } q \},$ such that $c_{axis} \neq \text{"attribute"}$
1.4	else if $(p_{axis} = q_{axis})$	$matchStep(p, q) \rightarrow matchChildren(p, q)$
1.5	else	$matchStep(p, q) \rightarrow False$
2	$matchChildren(p, q)$	
2.1	if $(p_{test} = \text{"*"} \vee (p_{test} = q_{test}))$	$matchChildren(p, q) \rightarrow matchPred(p_{pred}, q) \wedge matchNext(p_{next}, q))$
2.2	else	$matchChildren(p, q) \rightarrow False$
3	$matchPred(p_{pred}, q)$	
3.1	if $(p_{pred} = \text{null})$	$matchPred(p_{pred}, q) \rightarrow True$
3.2	else if $(q = \text{null})$	$matchPred(p_{pred}, q) \rightarrow False$
3.3	else if $(p_{pred} = p_1 \wedge p_2)$	$matchPred(p_{pred}, q) \rightarrow matchPred(p_1, q) \wedge matchPred(p_2, q)$
3.4	else if $(p_{pred} = p_1 \vee p_2)$	$matchPred(p_{pred}, q) \rightarrow matchPred(p_1, q) \vee matchPred(p_2, q)$
3.5	else	$matchPred(p_{pred}, q) \rightarrow matchStep(p_{pred}, q_{pred}) \vee matchStep(p_{pred}, q_{next})$
4	$matchNext(p_{next}, q)$	
4.1	if $(p_{next} = \text{null})$	$matchNext(p_{next}, q) \rightarrow True$
4.2	else if $(q = \text{null})$	$matchNext(p_{next}, q) \rightarrow False$
4.3	else	$matchNext(p_{next}, q) \rightarrow matchStep(p_{next}, q_{pred}) \vee matchStep(p_{next}, q_{next})$

FIG. 4

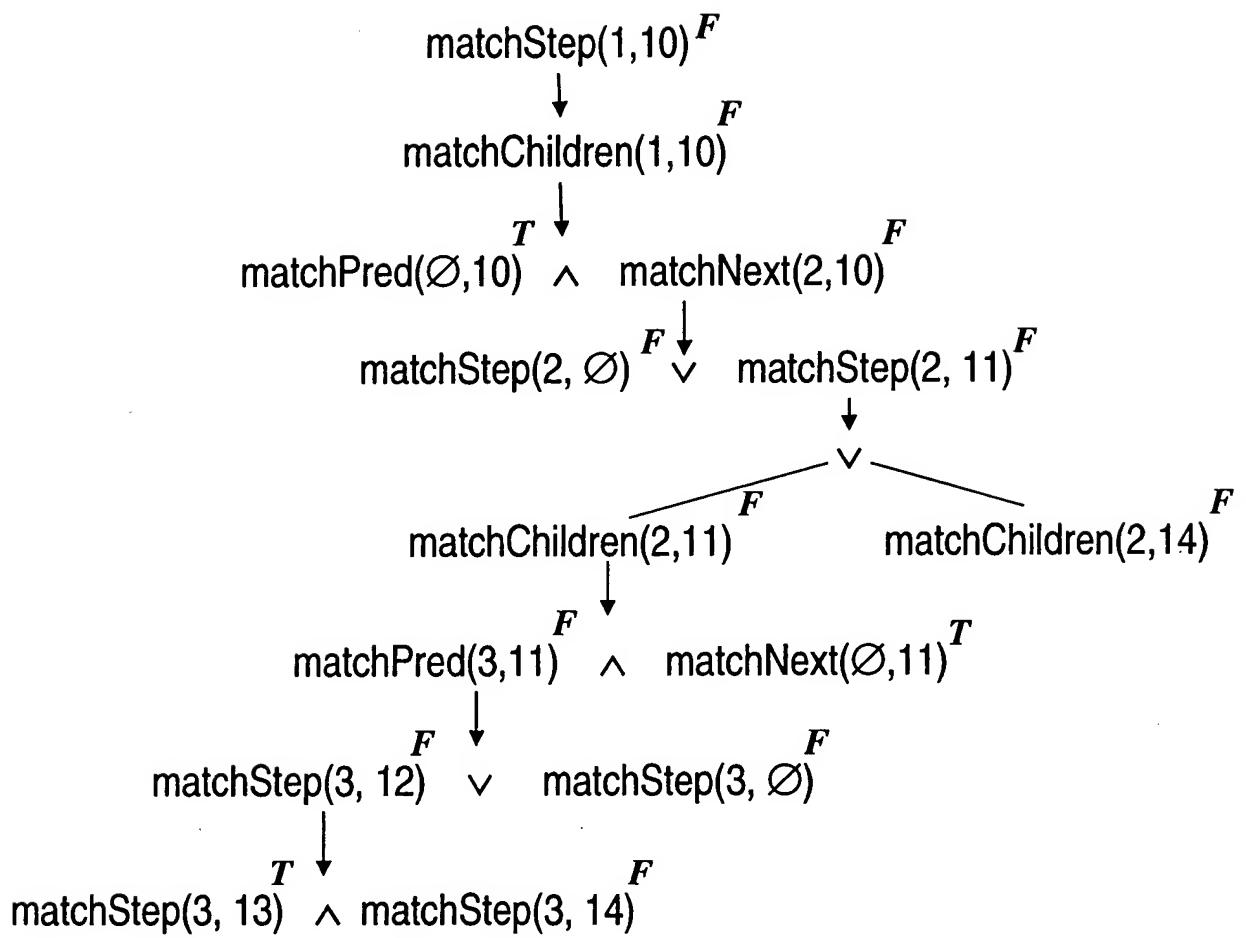
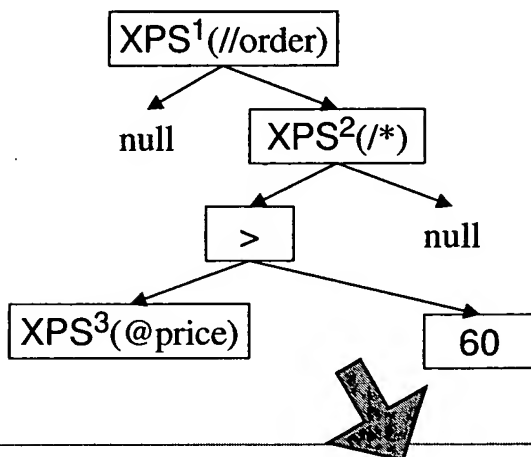
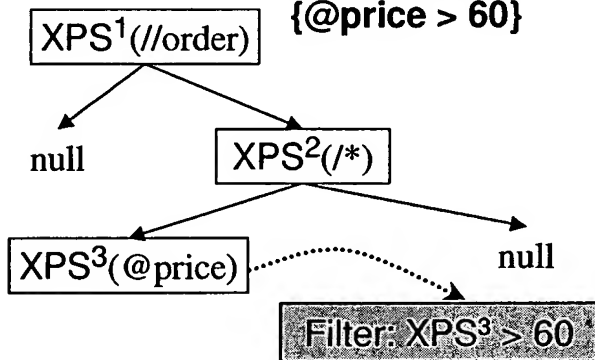


FIG. 5

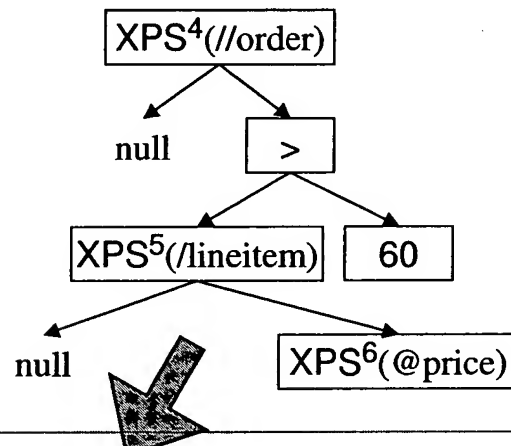
P = //order/*[@price > 60]



**P' = //order/*[@price]],
{@price > 60}**



Q = //order[lineitem/@price>100]



**Q' = //order[lineitem/@price],
{@price > 100}**

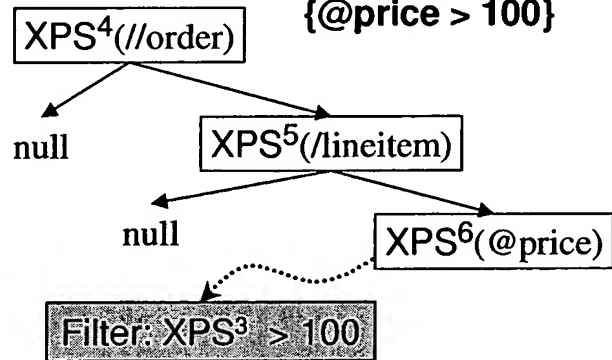


FIG. 6

ExtractPredicates(E)

```
foreach c in breadth-first traversal of E do  
    if ( $c \in \{<, <=, >, >=, =, \neq\}$ ) then  
        // left extraction point  
        lep = c.left;  
        if (lep is an XPS) // traverse next's  
            while (lep.next  $\neq$  null) lep=lep.next;  
        // right extraction point  
        rep=c.right;  
        if (rep is an PS) // traverse next's  
            while (rep.next  $\neq$  null) rep=rep.next;  
        add "c (lep, rep)" to the filter list  
        if (lep is a const)  $\wedge$  (rep is an XPS)  
            replace c with the right child  
        if (rep is a const)  $\wedge$  (lep is an XPS)  
            replace c with left child  
        if (lep is an XPS)  $\wedge$  (rep is an XPS)  
            replace comparison in c with AND
```

FIG. 7

P = //employee[.//employee/@salary
< .//employee/@bonus]//employee

Q = //employee[employee/@salary
< employee[employee/@salary <
employee/@bonus]//employee]

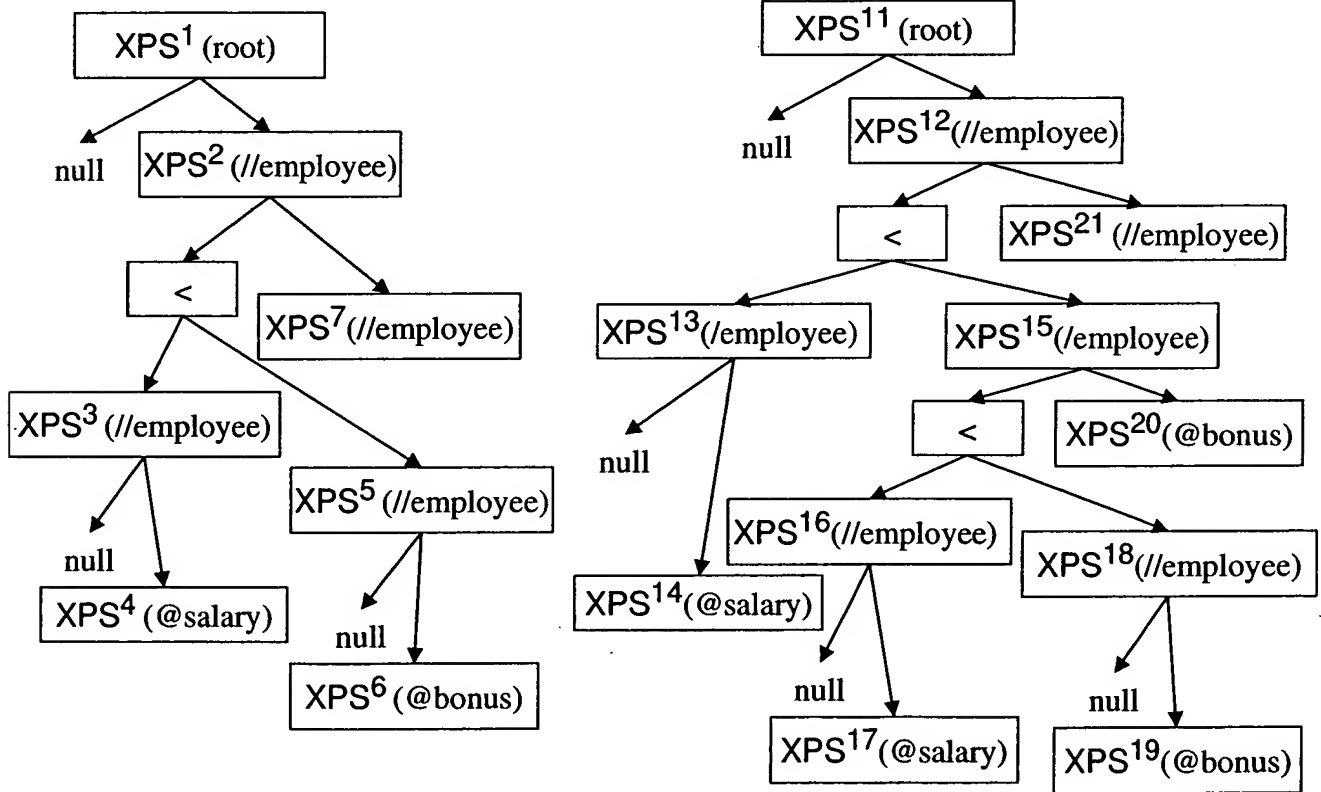


FIG. 8

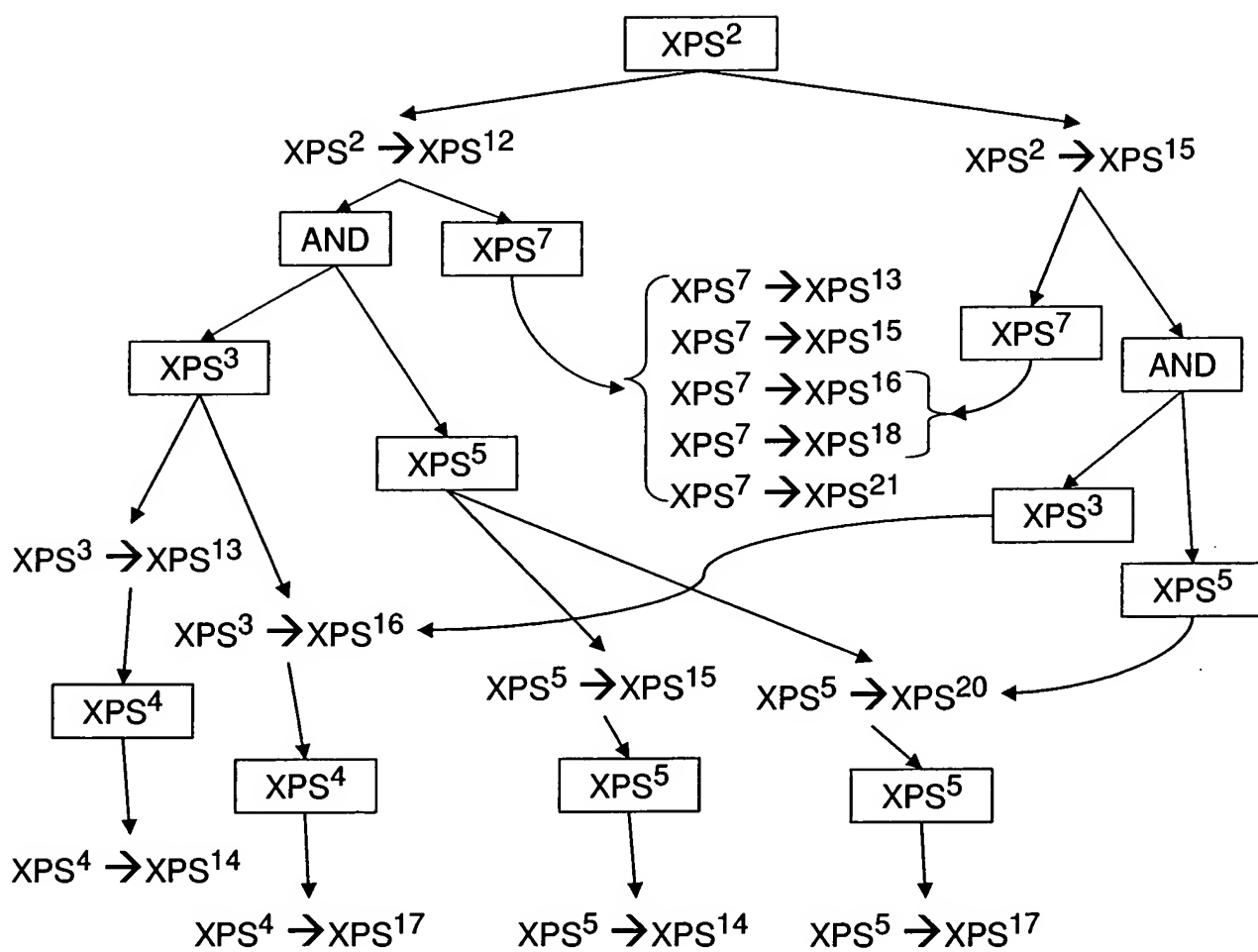


FIG. 9

REF/VALUE	PATH	TYPED VALUE
REF1	/A/B/C	5
REF2	/A/C	6
...

FIG. 10

<i>Entry</i>	<i>PXR content</i>	<i>Extra Navigation</i>	<i>Predicates</i>	<i>Name/Ancestors</i>
1	(NR)	Forward/Reverse	As residual	As residual
2	(DR)	Full Query	Full Query	Full Query
3	(copy)	Forward	Below QEP, as residual	No
4	(NR, path)	Forward/Reverse	As residual	As pushdown
5	(DR, path)	Full Query	Full Query	As pushdown
6	(copy, path)	Forward	Below QEP, as residual	As pushdown
7	(NR, value)	Forward/Reverse	As pushdown	As residual
8	(DR, value)	Full Query	As pushdown	Full Query
9	(copy, value)	Forward	As pushdown	Below QEP, as residual
10	(NR, path, value)	Forward/Reverse	As pushdown	As pushdown
11	(DR, path, value)	Full Query	Full Query	As pushdown
12	(copy, path, value)	Forward	As pushdown	As pushdown

FIG. 11

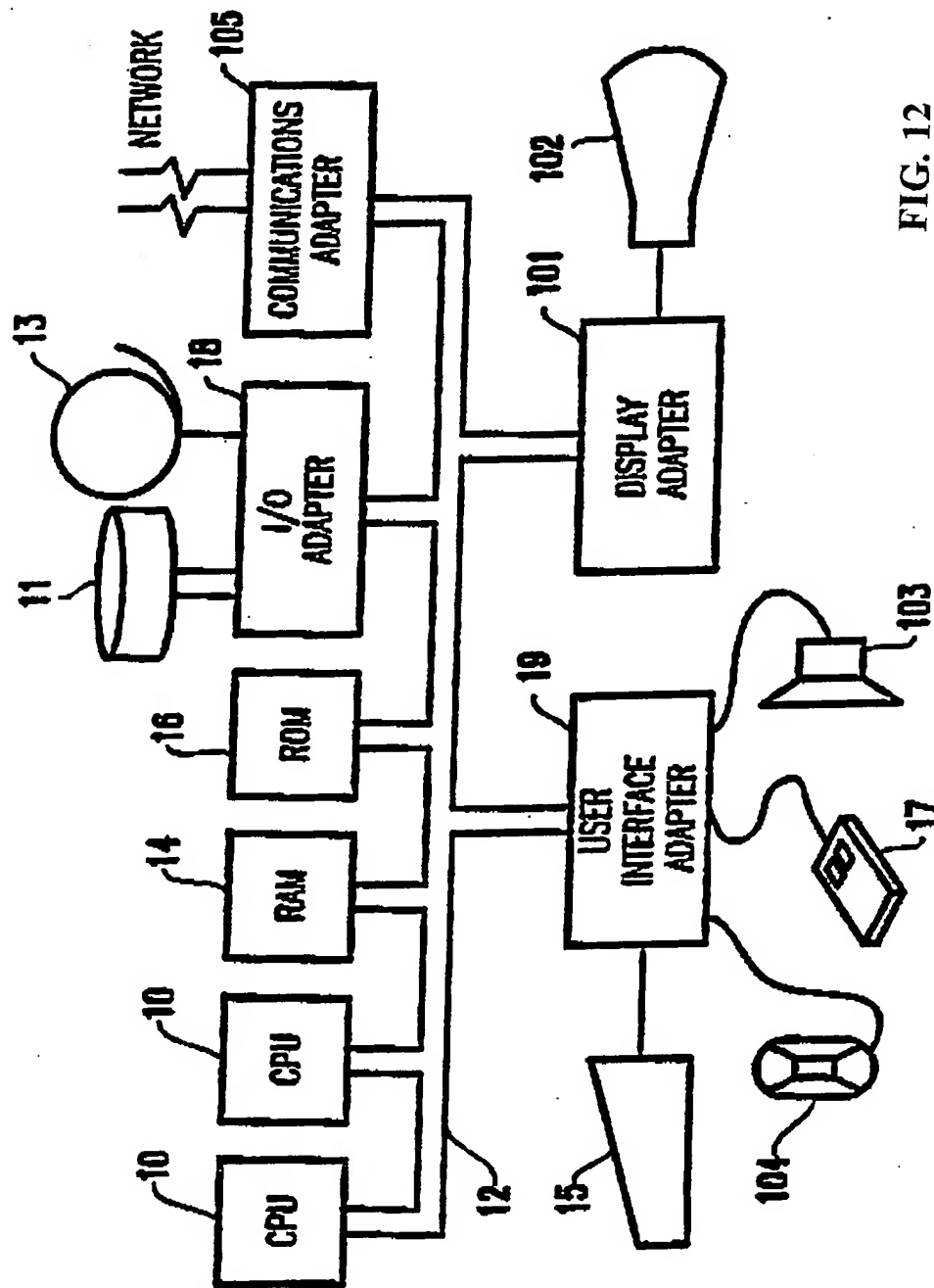


FIG. 12